UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/663,455 | 09/15/2003 | David Abu Ghazaleh | RSW920030055US1 | 2196 |

46270 7590 11/18/2008
(SAUL-RSW) PATENT DOCKETING CLERK
IBM Corporation (SAUL-RSW) C/O Saul Ewing LLP
Penn National Insurance Tower
2 North Second Street, 7th Floor
Harrisburg, PA 17101

| EXAMINER | |
|---|---|
| VU, TUAN A | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2193 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 11/18/2008 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/663,455 | GHAZALEH ET AL. | |
| | Examiner | Art Unit | |
| | TUAN A. VU | 2193 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *19 August 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-35* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

## DETAILED ACTION

1. This action is responsive to the Applicant's response filed 8/19/08.

As indicated in Applicant's response, claims 2, 14, 22-25, 28 have been amended. Claims 1-35 are pending in the office action.

### *Claim Rejections - 35 USC § 103*

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-2, 5-22, 24 and 26-35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bailey (US Pat. No. 6,701,513) in view of Kodosky (US PG-Pub. No. 2003/0184580).

**As per claim 1**, Bailey teaches a method for graphically representing object oriented programming logic (Abstract lines 10-13; *objects ... predefined properties... methods* -col. 4 lines 10-28), the method comprising the steps of:

(1) providing a plurality of different symbols for use in a diagram of object oriented programming logic, each different symbol representing a different type of object in object oriented programming (e.g. 402, 402a fig. 4A; 430a, 440, 436a – Fig. 4D; *object members ... class methods ... object classes* - col. 7, lines 57-61; Col. 8, lines 24- 26);

(2) selecting an object of the logic to be represented in the diagram (e.g. Form 1 object 404 – Fig. 4D; col. 10 lines 36-40, 53-55)

(3) labeling the symbol with a label descriptive of the object's features so that it is distinguishable from other symbols of the same object type (e.g. col. 10, line 53 to col. 11 line 2; col. 11, lines 35-38; 426a 434a – Fig. 4D);

(4) for each object assigned to or defined within said selected object, drawing a symbol corresponding to that object and labeling the symbol with a label descriptive of the object's features (e.g. col. 10, lines 53-55, 58-59, 64-65; col. 11, lines 35-38; 430a, 426a, 434a - Fig. 4D); and

(5) drawing a line between each object drawn in step (4) and another object in the graphical representation to which it is assigned or within which it is defined (e.g. col. 4, lines 23-26; link 440 – Fig. 4D; 426, 434 -Fig. 8A; *graphically linking* - col. 13 lines 8-20).

However, Bailey does not explicitly teach selecting an object in (2) as a *main object* of the logic to be represented in the diagram and drawing a symbol corresponding to the main object. Kodosky in analogous art teaches creating an icon (e.g. paragraph [0011] lines 10- 12) representing a main program in a hierarchical view of the system (e.g. paragraph [0012] lines 9-11, 24-25). It would have been obvious to one of ordinary skill in the art at the time of the invention was made to combine the teachings of Bailey and Kodosky because Kodosky's teaching of main object with symbol would help user to easily understand logic that was described in detail according to specification of the system without creating any confusion.

**As per claim 2**, Bailey teaches the method, further comprising the step of: (6) providing a plurality of additional different symbols for use in the diagram, each of the additional different symbols representing a different an object oriented programming element other than an object (e.g. col. 4, lines 23-26).

**As per claim 5**, Bailey teaches the method, wherein the labels comprise text (e.g. col. 10, lines 53-55).

**As per claim 6**, Bailey teaches the method, wherein step (5) comprises drawing the line between the object defined in step (4) and another object it is most directly assigned to or is most directly defined within (e.g. col. 4, lines 36-40).

**As per claims 7 and 8**, Kodosky teaches a method of visually creating a distributed system design and software programming which can be used in documenting software and to prepare a program specification (e.g, see *Print documentation* – Fig. 20A). One would be motivated to enable Bailey's framework building instance to be documented and distributed for further builds, data acquisition and testing (as suggested in Bailey: col. 3 lines 24-36)

**As per claim 9**, Bailey teaches the method, further comprising the step of: (8) repeating steps (1) - (5) to prepare a plurality of separate diagrams corresponding to separate parts of an overall application (e.g. col. 8, lines 14-18).

However, does not explicitly teach a first object is the main object appearing in at least a first one of the diagrams and is not a main object appearing in at least a second one of the diagrams.

Kodosky teaches a first object is the main object appearing in at least a first one of the diagrams and is not a main object appearing in at least a second one of the diagrams (e.g. paragraph [0015]). One would have been motivated to use Kodosky's main object to trigger the graphical enlistment of further objects in Bailey's endeavor, based on the connecting of icons as set forth in claim 1.

**As per claim 10**, Bailey does not disclose wherein the second diagram does not disclose objects assigned to and defined within the first object and the first diagram does disclose objects assigned to and defined within the first object. Based on the hierarchy of OO objects in Kodosky, and the well-known concept that base class include members defined therein in set forth in Baileys, the above submember of a base class would have flow out of the OOP approach in both Bailey (refer to claim 1) and Kodosky. Enabling a base class to disclose its defined member as opposed to the other way around (e.g. the second object not disclosing base/first object it is assigned to) would have been a obvious feature adopted in Bailey (in light of Kodosky)

**As per claim 11**, Bailey teaches the method, an application-level representation disclosing an overall software system (e.g. col. 8 lines 14-18)

**As per claim 12**, Bailey does not explicitly disclose wherein label of the first object in the second diagram identifies the first diagram as further disclosing details of the first object. Bailey teaches object-oriented enlistment of base objects prior to expanding the base with submembers wherein the label identifies as disclosing further details of the object (e.g. col. 9, lines 50-53). Enabling object in graphical representation of a base class to identifies itself even when such identification is shown in a base class or submember would have been obvious; and this is evidenced with OO nomenclature where a subclass (second diagram) has a label identifying a parent class, separated by a "." (label of a first object). Bailey teaches object-oriented building with labeling using this nomenclature (e.g. Label1.Caption, Form1.Label1 Fig. 8B), it would have been obvious for one skill in the art at the time the invention was made to implement the OOP by Bailey (in view of Kodosky) so that labeling would adopt this OO

nomenclature such that the first class or object is represented in the subclass or second object representation, i.e. the base object identifies itself in the labeling of the sub class.

**As per claim 13**, Bailey does not explicitly teach wherein the symbols representing different object types include: a symbol for representing objects that are application type objects; a symbol for representing objects that are window type objects; a symbol for representing objects that are class type objects; a symbol for representing objects that are event script type objects; and a symbol for representing objects that are method type objects. Bailey does teach plurality of objects or icons representing text boxes, radio buttons, scroll bars, menu bars and so on (e.g. col. 2, lines 3-9; col. 7, lines 57-61; col. 8, lines 8-11, 15-18, 24-37; Form.Label1, Form.Break1, Form.GreaterThan1, Form.CBAdd1, Form.TextBox - Fig. 14B-C; Form1.VSrollbar1 Form1.Label1 – Fig. 4D – Note: Form1.xxx is analogized to even script type; Form reads on script type of symbols, and any action associated thereto reads on event). Each icon represents a corresponding object class that is available for use by developer. It would have been obvious to one of ordinary skill in the art at the time of the invention was to add more graphical icons representing other programming objects which will improve the functionality of the system.

**As per claim 14**, it is similar to claim 13 with added symbols; therefore, it is rejected for the same rationale as claim 13; and further Bailey teach (i) data transfer type and (ii) inheritance type symbols ((i) Form.Wait1, Form.User1; Form.Yield1 – Fig. 16; (ii) Form1.Variable1; Form1.CommandButton1 – Fig. 14A ). Bailey disclose remote queries of COM objects (col 24 li. 48-65), hence it would have been obvious for one skill in the art at the time the invention was made to provide symbols denoting database type and remote link type because objects retrieved

from a COM paradigm (Bailey: col. 8 lines 37-43) in light of the factory of objects (Bailey: col 7 lines 61-66) would necessitate links and database identifications.

**As per claim 15**, Bailey teaches the method, wherein the sixth, eighth, and ninth symbols are drawn connecting two other object symbols (e.g. col. 4, lines 35-40).

**As per claim 16**, it is similar to claim 13 with added symbols; therefore it is rejected for the same rationale as claim 13.

**As per claim 17**, Bailey teaches the method, further comprising the step of: providing in a separate description of the logic to be performed responsive to an event script (e.g. col. 10, lines 9-12).

**As per claim 18**, Bailey teaches the method, wherein the symbol representing event script type objects is drawn connected to another object that directly executes the event script corresponding to the event script symbol (e.g. Form1.VSrollbar1 ←> Form1.Label1 – Fig. 4D).

**As per claim 19**, Bailey does not explicitly teach wherein the symbol representing method type objects is drawn connected to the main object of the diagram and represents that the object is available within that main object and does not represent that the main object invokes it. Kodosky teaches objects with line to point to the underlying methods (Stop 846, Shut-down 850 – Fig. 35, 28A ) and main program symbols drawn connected to subprograms to execute functions described to be enclosed for the base object (para 0314). It would have been obvious for one skill in the art at the time the invention was made to implement visual representation in Bailey, so that base object can be represented as to be linked to underlying sub-program or method icons as set forth above, as for informing the developer that some base object can offer by its presence the availability of its underlying methods or subprograms (not for the base object

to actually invoke them). One would be motivated to do so because such representation would enable the developer in Bailey to have prior knowledge about what base objects acquire in order to further implement from core functions whereby reach a predefined work flow, which is exactly data acquisition by Bailey (see Function, Core, Data Acquisition, Logic : Fig 11, 13)

**As per claim 20**, Bailey teaches the method, wherein the method is implemented via a computer program, and wherein step (1) comprises providing a graphical user interface in which a user is presented with a pallet containing the symbols (e.g. col. 7, lines 57-61 ). However, Bailey does not explicitly teach wherein steps (3) and (4) comprise dragging and dropping the symbols from the pallet into a work area. Analogous to Bailey's mention of well-known tools s where drag and drop (e.g. Drag – col. 11 lines 36-44) are common practices (col. 2 lines 20-30) such as in Visual Basic COM technologies, Kodosky in an analogous endeavor, teaches a design tool that enables the user to drag and drop symbols from the pallet into a workspace (e.g. paragraph [0037]; [0034] lines 6-9). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention was made to combine the teachings of object-oriented manual manipulation in Bailey's tool using this well-known practice of drag-and-drop (evidenced in Kodosky) so that user or developer can easily visualize, select and manually join, add, and/or deploy iconic objects for implementing the logic or application based on the availability of various visual GUI components in a object-based visual developing tool, just as this drag-and-drop practice was practiced for its benefits by many visual developments technologies as set forth above, at the time the invention was made.

**As per claim 21**, Bailey teaches the method, wherein the method is implemented via a computer program, and wherein step (1) comprises providing a graphical user interface in which

a user is presented with a pallet containing the symbols (e.g. col. 7, lines 57-61) and wherein steps (3) and (4) comprise dragging and dropping the symbols from

the pallet into a work area (e.g. col. 12, lines 35-37), and wherein the labels comprise text (e.g. col. 10, lines 53-55) and further wherein at least some of the text labels can be made to appear in the graphical representation via an action taken by a user (e.g. col. 10, lines 63- 65).

**As per claim 22**, Bailey teaches the invention substantially as claimed including a computer readable product embodied on computer readable media readable by a computing device for enabling a user to generate a graphical representation of object oriented programming logic (Abstract lines 10-13; col. 3, lines 24-27; col. 7, lines 57-61; refer to claim 1), the product comprising executable instructions that:

(i) provide a graphical user interface in which a user is presented with a plurality of different symbols for use in developing a graphical representation of object oriented programming logic, each different symbol representing a different type of object in object oriented programming (e.g. col. 7, lines 57-61; 402, 402a fig. 4A; 430a, 440, 436a – Fig. 4D; *object members ... class methods ... object classes* - col. 7, lines 57-61; Col. 8, lines 24- 26);

(ii) enable the user to select an object of the logic to be represented in the diagram (e.g. Form 1 object 404 – Fig. 4D; col. 10 lines 36-40, 53-55);

(iii) enable the user to draw a symbol corresponding to that selected object and label the symbol with a label descriptive of the object's features (e.g. col. 10, line 53 to col. 11 line 2; col. 11, lines 35-38; 430a, 426a, 434a - Fig. 4D) so that it is distinguishable from other symbols of the same object type (Note: a label for one object teaches label for a distinguishing purpose, e.g. label 1, label 2, form 1, form 2 - see Fig. 4A-D);

(iv) enable the user to label the symbols with a label descriptive of the object's feature ( see above);

(v) enable the user to draw lines between symbols in the workspace (e.g. col. 4, lines 35-40; col. 9, lines 60-62; e.g. col. 4, lines 23-26; link 440 – Fig. 4D; 426, 434 -Fig. 8A; *graphically linking* - col. 13 lines 8-20), the line between each symbol and another object in the workspace to which it is assigned and in which it is defined (see Fig. 4A-B; 430a, 440, 436a – Fig. 4D).

However, Bailey does not explicitly teach (iv) instructions such to enable the user to drag and drop symbols; nor does Bailey teach (iv) to draw a line between a symbol and another object such that the symbol and the another object are *dragged and dropped* into the workspace. But this 'drag-and-drop' limitation has been addressed in claim 20.

Nor does Bailey teach that the selected object is selecting as *main object.* This limitation has been rendered obvious in claim 1.

**As per claim 24**, Bailey does not teach denote a main object; but Kodosky teaches, computer executable instructions (ii) that enable the user to denote one and only one object in the workspace as a main object (e.g. paragraph [0012] lines 9-11,24-25). This *denote* limitation would have been obvious in light of Kodosky and the corresponding rationale regarding 'main object' as set forth in claim 1.

**As per claim 26**, Bailey teaches the method, wherein the labels are text labels (e.g. col. 10, lines 53-55).

**As per claim 27**, Bailey teaches the method, further comprising: computer readable instructions that enable the user to prepare a plurality of the diagrams corresponding to separate parts of an overall application and further comprising computer readable instructions for

enabling the user to specify relationships between individual ones of the diagrams (e.g. col. 8, lines 14-18; col. 9, lines 53-57; col. 16 line 5 to col 17 line 18; Fig. 14A-D).

**As per claim 28**, Bailey teaches, wherein the sixth computer readable instructions comprise instructions that enable the user to include references associated with symbols in one diagram identifying at least one other diagram within which the object represented by that symbol also appears (e.g Fig. 14A-D and related text).

**As per claim 29**, Bailey does not explicitly disclose relationships between object of first diagram and second diagram wherein: (1) the second diagram discloses additional details about the object in the first diagram; (2) the second diagram shows the object in a more abstract context than the first diagram and (3) the object is the main object of the second diagram. But the object-oriented approach in Bailey (refer to claim 1) has been further evidenced with hierarchy of objects as in Kodosky whereby the OO relationships between the hierarchical object (as represented in the first and second diagrams) are selected from the group comprising of base objects and sub-objects. Based on the well-known concept of OO methodology, the limitations (1) where a subclass exposes some reference regarding its base class;(2) where the notation of a subclass shows global (non-detailed) reference of its base class nomenclature; and (3) where the base class is depicted in the diagram notation or representation of a subclass, would all have been obvious in view of the rationale of claims 9 and 10, wherein the diagram representing a derived or sub object not fully showing the internals of its parent object which it has been sub related to or derived from.

**As per claim 30**, Bailey does not explicitly teach wherein the symbols representing different object types include: a symbol for representing objects that are application type objects;

a symbol for representing objects that are window type objects; a symbol for representing objects that are class type objects; a symbol for representing

objects that are event script type objects; and a symbol for representing objects that are method type objects.

Bailey does teach plurality of objects or icons representing text boxes, radio buttons, scroll bars, menu bars and so on (e.g. col. 2, lines 3-9; col. 7, lines 57-61; col. 8, lines 8-11, 15-18, 24-37). Each icon represents a corresponding object class that is available for use by developer. It would have been obvious to one of ordinary skill in the art at the time of the invention was to add more graphical icons representing other programming objects which will improve the functionality of the system.

**As per claim 31**, Bailey does not explicitly teach the method, wherein the symbols representing different object types include: a first symbol for representing objects that are application type objects; a second symbol for representing objects that are window type objects; a third symbol for representing objects that are class type objects; a fourth symbol for representing objects that are event script type objects; and a fifth symbol for representing objects that are method type objects; and wherein the additional symbols representing additional program elements include: a sixth symbol for representing data transfers; a seventh symbol for representing databases; an eighth symbol for representing remote links; and a ninth symbol for representing inheritance.

Bailey does teach plurality of objects or icons representing text boxes, radio buttons, scroll bars, menu bars and so on (e.g. col. 2, lines 3-9; col. 7, lines 57-61; col. 8, lines 8-11, 15-18, 24-37). Each icon represents a corresponding object class that is available for use by

developer. It would have been obvious to one of ordinary skill in the art at the time of the invention was to add more graphical icons representing other programming objects which will improve the functionality of the system.

**As per claim 32**, Bailey teaches the method, seventh computer executable instructions that restrict the user to using the sixth, eighth, and ninth symbols to connect two other object symbols (e.g. col. 4, lines 35-40).

**As per claim 33**, it is similar to claim 30 with added symbols; therefore, it is rejected for the same rationale as claim 30.

**As per claim 34**, Bailey teaches the method, further comprising:
computer executable instructions that enable the user to providing in a separate description of the logic to be performed responsive to an event script (e.g. col. 9, lines 9-12).

**As per claim 35**, Bailey teaches the method, further comprising: computer executable instructions that enable the user to insert text associated with symbols in the workspace that can be made to appear in the workspace responsive to an action taken by a user (e.g. col. 10, lines 9-12, 53-55, 58-59, 64-65; col. 34, lines 16-20).

4. Claims 3, 4, 23 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bailey (US Pat. No. 6,701,513) in view of Kodosky (US PG-Pub. No. 2003/0184580) further in view of Visio 2000 Standard Edition User Guide (Published by Visio International 1999 hereinafter Visio).

**As per claim 3**, Kodosky teaches the method, further comprising the step of: graphically denoting the main object in the diagram (e.g. paragraph [0012] lines 9-11, 24- 25); but combined

with Bailey, does explicitly disclose drawing another symbol around the symbol for the main object.

Visio teaches drawing another symbol around the symbol for the main object (pages 15-17). Based on a visual tool wherein highlighting of an object of interest or icon from which to draw more connected associations in Bailey (see Fig. 4A-D) and drawing a square-shape outline around an object for easy focus, it would have been obvious to one of ordinary skill in the art at the time of the invention was made to combine the teachings of Kodosky (in light of Bailey) and Visio such that user or developer can easily create, distribute and/or deploy application with identifying various components in a distributed system as a main component in the system.

**As per claim 4**, Visio teaches the method, wherein step (7) comprises drawing a circle completely enclosing the symbol of the main object (page 18, see shape-to-shape connections).

**As per claims 23 and 25**, they are the product claims of claims 3 and 4, therefore; they are rejected for the same reason as per claims 3 and 4 above.

***Response to Arguments***

5. Applicant's arguments filed 8/19/08 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

**35 USC § 103(a) Rejection**:

(A) Applicants have submitted that for claim 1, the work flow diagram of Bailey does not teach drawing a line between objects to represent assignment to or definition within another object of the OOP -- object-oriented programming, a concept which Bailey does not address (Appl. Rmrks pg. 16, top 2 para). Bailey teaches joining symbols representing objects to define a functionality that interrelates the objects (see Fig. 4A-D) and there is clear object-oriented

factory-based building wherein object classes with their associated members, pre-defined properties and methods are fetched/instantiated as from an available object factory in Bailey's visual tool (see col. 7 line 53-66; col. 4 lines 10-28), which renders the argument about a mere 'work flow diagram' misplaced. There is nothing compelling as requirement in the language (emphasis added) recited as "line between each object … and another object in the … graphical representation *to which it is assigned* or *within which it is defined*" so that such language clearly precludes the linking of graphical icons by Bailey – drawing of line to denote their functional dependency (see Fig. 4D, 8D) in terms of their point of junction or nature thereof (e.g. *terminals … sink object … source object* - col. 16 lines 5-31) – from reading onto this 'representation to which it is assigned or within which it is defined', which appears to be a broad relationship relating to a 'graphical representation'. Iconic representations joined one another by a link for depicting association with another iconic representation in an OOP signifies designer-based (a) visual assignment of an entity to a graphical representation of other object/entity, (b) definition of an entity within the scope of another graphical representation. Thus, objects drawn with lines to join them (see Fig 4, 8) in a visual tool like in Bailey's OOP approach using factory of predefined object class and methods entail a development stage wherein any hand-linking by the developer is for assigning a symbol property or iconic functionality to another symbol representing some object(s) initially created by the designer in the visual enlistment palette/pane as shown in the rejection. Specifically, the very act of joining two symbols with specific bubble to denote some form of terminals suffice to meet 'to which it is assigned or within which it is defined', with 'it' being either a symbol, a icon, or a class object. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define

a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference.

(B) Applicants have submitted that relying of Kodosky for a 'main object' seems to be a impermissible hindsight approach because there is no suggestion to combine a non OOP by Bailey to Kodosky's object hierarchical design (Appl. Rmrks pg. 16 bottom). The teachings used in the rejection flow out from the references, no remote teaching is gleaned or borrowed from the current Disclosure. That is, it is shown that object-oriented design basis has been deemed integral in Bailey's development environment and it is deemed all the more reasonable to combine Kodosky's hierarchy of objects (so as to designate a top *main object* as for starting a work flow) with the objects linking within a flow design approach by Bailey. Applicants fail to point out via factual evidences how combining the hierarchy with top down objects by Kodosky would demise the work flow builder using object classes and predefined factory in Bailey. The hindsight argument is deemed non-persuasive.

(C ) Applicants have submitted that for claim 3, there is nothing in Visio to the effect of 'drawing a circle around a main object to denote it as … main object of … representation of object-oriented programming' (Appl. Rmrks pg. 17, bottom). The 'denoting' of a main object is a necessary step in Bailey, because in a framework design where a base object is instantiated by user selection whereby additional objects are linked or added to associate with this main object, this starting step is taught in Bailey otherwise the process of expanding the start object with additional members, objects (i.e. as cited in Bailey: col. 10 lines 36-40, 53-55) would not have been able to go further. Double-clicking or highlighting this object would have been analogized to 'denoting', and manually effectuating a focus-type of event (i.e. user actions) in visual design

(such as in Bailey) where mouse click/movement is preponderant suggest circling the selected or highlighted item. A peripheral line of circular or rectangular shape drawn at the contour of one object of interest (see Kodosky) to enable it to be more visible for further action has been further exemplified in contours by Visio. That is, highlighting an icon by way of a mouse click (see Bailey: col. 13 middle) would have been another of such means to drawn a contour, whether this is circular or rectangular, for easy focus as explained above. There is sufficient teaching in Bailey's manual selecting and enlisting of objects on a visual frame, so that combined with the hierarchy of objects (as in Kodosky's having a base object atop), and the contours set forth in Visio, the 'denoting' limitation in terms of a drawing circular contour (around a base object) would have been an obvious limitation. The Applicants fail to point out -- via factual evidences to the contrary - how using a manual clicking of mouse in Bailey in light of a base hierarchical object in Kodosky combined with the highlighting (square contour) in Visio, fail to teach the 'denoting' as recited, such that the combination as proffered would have been non-obvious. The argument is not persuasive.

(D) Applicants have submitted that Kodosky does not teach 'document software' and the 'plurality of diagrams … separate parts … first object … appearing … first one of the diagrams …' (Appl. Rmrks pg. 18-19). These allegations are referred back to the responses regarding 'drawing a line' among symbols and the use of base object to start enlisting further object oriented as set forth in sections B and C. Object-oriented building where a base object is a start for object/class members or subclasses (including properties and methods thereof) to be further added to has been repeatedly cited throughout the rejection (refer to claim 1); that is claim 9 is deemed obvious. As of claim 7, the documentation of one building instance for further usage to

a more distributed manner is disclosed in Kodosky. The rationale to render obvious claim 7 is deemed reasonable in light of Bailey's enterprise-wide methodologies; and the arguments are not persuasive.

(E) Applicants have submitted that para 0015 of Kodosky is irrelevant to claim 10 (Appl. Rmrks pg. 19). The rationale to render claim 10 is now readjusted to render the allegations misplaced.

(F) Applicants have submitted that Bailey's col. 9 does not match claim 12 (Appl. Rmrks pg 20-21) reciting of 'labels of the first object in the second diagram … identifies the first diagram'. The argument is not commensurate with the actual grounds of rejection.

(G) Applicants have submitted that for claims 13, 14, 16 Bailey fails to teach 'symbols representing application-type, window-type … method-type objects' (Appl. Rmrks pg. 21-22) particularly in view of the current Disclosure. The claims 13-14 recite 'wherein the symbols … object types include" hence does not enforce that each and every object symbol type to be matched with Bailey. Nevertheless, Bailey is shown to disclose some class type, window type, method type, additional element type. The language of the claim regarding 'include' cannot enforce a 'consist of' requirement. The Specifications cannot be read into the claim. The argument is deemed insufficient to overcome what has been cited in the rejection, based on the rationale as to render some of the symbol types obvious; rendering claim 15 obvious as a result of failure of Applicants to point out how the rationale in claim 14 would be deficient.

(H) Applicants have submitted that for claim 18, Bailey fails to disclose event script type (Appl. Rmrks pg. 24) because the Office action misinterprets event script object connected to another same object. The cited Form1 joined by a line to effectuate action in terms of how one

Form interrelate functionally with the other discloses 'executes the event script corresponding to the … event script symbol' as recited, wherein event script symbol are identified in claim 13.

(I) Applicants have submitted that for claim 19, Bailey as cited does not teach a line to represent that the main object is available (Appl. Rmrks pg. 24 bottom, pg. 25). The rejection as now in effect does not appear to be commensurate with the above allegation.

(J) Applicants have submitted that claim 22 is recited with matter of claim 1, such that claims 25, 27, 28 regarding linked diagrams as part of overall application are not properly disclosed by Bailey (Appl. Rmrks pg 26-27). Claim 1 has been addressed in the above sections. As for the subject matter of claims 27-28, the cited parts disclose how to accomplish a work flow as contemplated in the visual tool by Bailey, i.e. objects are manually clicked or dragged into a pane, including their properties or methods retrieved from factory of objects as well as their pertinent properties (refer to Bailey: all the text depicting Fig. 4A-D, 8, 14-15). Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference; i.e. exactly via comparison of teachings therein with respect to the very constructs of the claim as in a one-on-one basis.

In all, the arguments are in part not commensurate with the specifics of the rejection as currently presented, and/or are otherwise not sufficient to overcome the rejection. The claims will stand rejected as set forth above.

***Conclusion***

6. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 ( for non-official correspondence - please consult Examiner before using) or 571-273-8300 ( for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.
Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished

applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

November 16, 2008